CS M151B Homework 5

Baolinh Nguyen
UID: 104732121

4.13 This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

```
add r5,r2,r1
lw  r3,4(r5)
lw  r2,0(r2)
or  r3,r5,r3
sw  r3,0(r5)
```

4.13.1 [5] <§4.7> If there is no forwarding or hazard detection, insert nops to ensure correct execution.

Q: how many cycles would it take to execute this code on the 5-stage pipelined datapath with forwarding logic?

4.13.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ADD	IF	ID	EX	M	WB									
NOP		IF	ID	EX	M	WB								
NOP			IF	ID	EX	M	WB							
LW				IF	ID	EX	M	WB						
LW					IF	ID	EX	M	WB					
NOP						IF	ID	EX	M	WB				
OR							IF	ID	EX	M	WB			
NOP								IF	ID	EX	M	WB		
NOP									IF	ID	EX	M	WB	
SW										IF	ID	EX	M	WB

14 cycles

4.13.1 (w/ forwarding)

	1	2	3	4	5	6	7	8	9
ADD	IF	ID	EX	M	WB				
LW		IF	ID	EX	M	WB			
LW			IF	ID	EX	M	WB		
OR				IF	ID	EX	M	WB	
SW					IF	ID	EX	M	WB

9 cycles

CS M151B Homework 5

Baolinh Nguyen

UID: 104732121

4.14 This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
label1  lw r2,0(r1)
~~label1~~: beq r2,r0,label2 # not taken once, then taken
        lw r3,0(r2)
        beq r3,r0,label1 # taken
        add r1,r3,r1
label2: sw r1,0(r2)
```

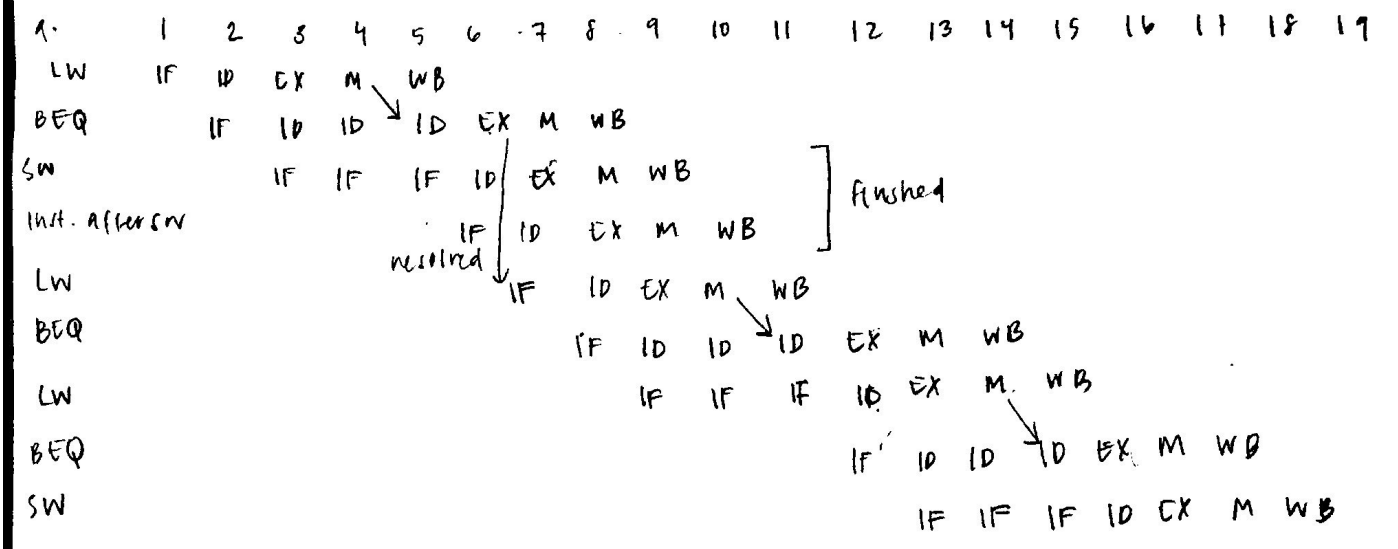
4.14.1 [10] <§4.8> Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

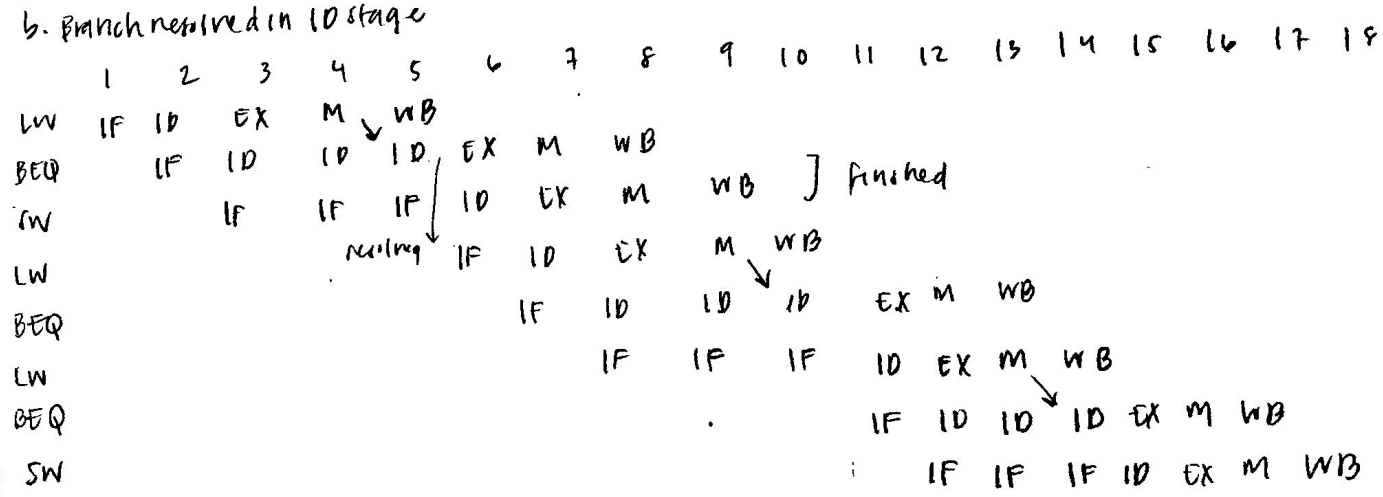
Show the pipeline diagram for the following cases:

a) full forwarding, branches resolved in EX

b) full forwarding, branches resolved in ID

4.14.1.





CS M151B Homework 5
Baolinh Nguyen
UID: 104732121

4.16 This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, NT, T, T, NT

4.16.1 [5] <§4.8> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

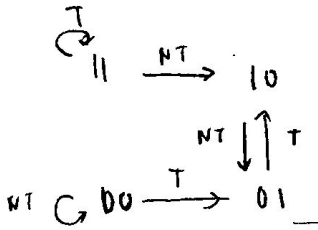
4.16.2 [5] <§4.8> What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.63 (predict not taken)?

4.16.3 [10] <§4.8> What is the accuracy of the two-bit predictor if this pattern is repeated forever?

4.16.1 T, NT, T, T, NT

Always-Taken: 3/5 Always-Not-Taken: 2/5

4.16.2



T, NT, T, T

Predictor	00	01	00	01
Result	X	✓	X	X

accuracy: 1/4

4.16.3 [T, NT, T, T, NT, T, NT, T, T, NT]

	T	NT	T	T	NT	T	NT	T	T	NT
Predictor	00	01	00	01	10	01	10	01	10	01
Result	X	✓	X	X	X	X	X	X	✓	X

1/5 accuracy